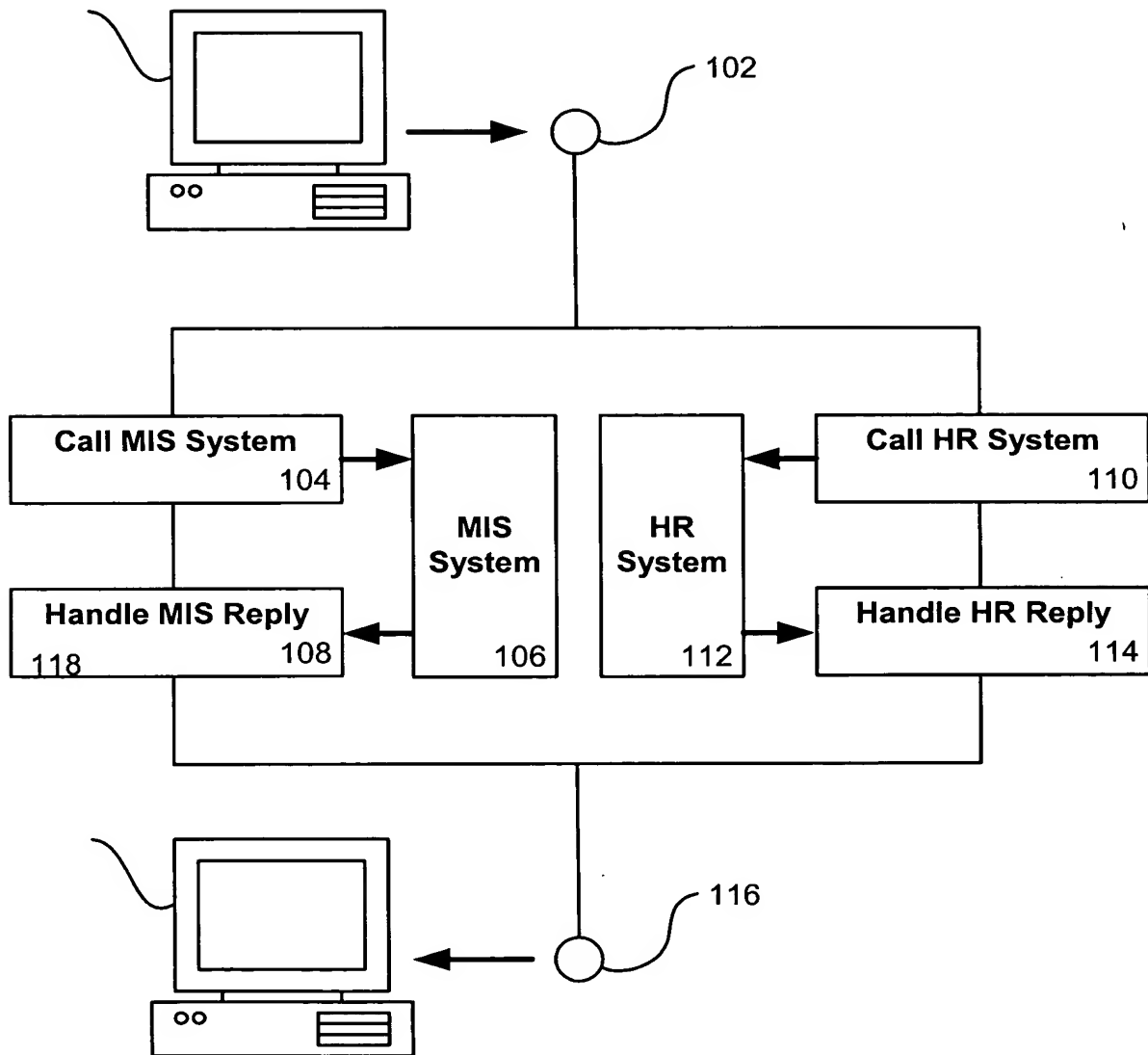


100



*Figure 1*

```
/**
 * @jwf:flow flow ::
 *
 * <process>
 *   <receive name="Receive employee info" method="start"/>
 *   <parallel join-condition="AND">
 *     <branch>
 *       <perform name="Request e-mail" method="reqMail"/>
 *       <receive name="Get e-mail" method="MIS_mailReady"/>
 *       <perform name="Notify client about e-mail"
 *         method="mailNotification"/>
 *     </branch>
 *     <branch>
 *       <perform name="Request benefits" method="reqBenefits"/>
 *       <receive name="Get benefits" method="hr_benefitsReady"/>
 *       <perform name="Notify client about benefits"
 *         method="benefitsNotification"/>
 *     </branch>
 *   </parallel> * <perform name="Reply to requestor" method="end"/>
 * </process>
 * ::
 **/
```

*Figure 2*

```

/**
 * @jwf:flow flow::
 *
 * <process name="PurchaseOrder">
 *   <receive name="Receive PO" method="getPO"/>
 *   <forEach name="processLineItems" var="lineitem"
 *     expression="getLineItems"
 *     parameters="inputPO">
 *     <perform name="Process line item" method="processOrder"/>
 *     <receive name="Handle service ack."
 *       method="orderService_sendAck"/>
 *   </forEach>
 *   <perform name="Send reply to the client" method="sendReply"/>
 * </process>
 * ::
 *
 * xquery::
 * define function getLineItems (element $po) returns element* {
 *   $po/DATAAREA/PROCESS_PO/POORDERLIN }
 * define function concat (element $x1, element $x2) returns element {
 *   $x1 + $x2 }
 * define function buildReply (element $x1) returns element {
 *   <reply>$x1</reply> }
 * ::
 */

public class PurchaseOrder {
    public XML inputPO;
    public XML lineitem;
    /**
     * @jwf:xml-sequence
     */
    public XML poAckList;
    /**
     * @jwf:transforms
     */
    PoTransforms transforms;
    /**
     * @jws:control
     */
    public OrderProcessor orderService;
    void getPO(XML po) {
        inputPo = po;
    }
    public void processOrder() {
        orderService.processOrder(lineitem);
    }
    public void orderService_sendAck(XML ackedLine)
        throws Exception {
        poAckList = transforms.concat(poAckList, ackedLine);
    }
    public void sendReply() {
        callback.reply(transforms.buildReply(poAckList));
    }
    public Callback callback;
    public interface Callback {
        public void reply(XML ack);
    }
}

```

*Figure 3*